UNIX™ **System V** — Release 2.0
Operator Guide
DEC™ **Processors**

April 1984
307-102, Issue 2

UNIX is a trademark of AT&T Bell Laboratories

DEC is a trademark of Digital Equipment Corporation

Copyright © 1984 AT&T Technologies
All Rights Reserved
Printed in U.S.A.

AT&T

CONTENTS

Chapter 1. INTRODUCTION

Chapter 2. START-UP PROCEDURES

Chapter 3. SINGLE USER AND MULTIUSER

Chapter 4. OPERATOR DUTIES

Chapter 5. VAX-11/750 OPERATIONS

Chapter 6. VAX-11/780 OPERATIONS

Chapter 7. PDP-11/70 OPERATIONS

Chapter 1

INTRODUCTION

This guide contains a description of console operations and step-by-step instructions for normal operator functions as they apply to the PDP*-11/70, VAX*-11/750, and VAX-11/780 processors. It is intended for UNIX† system operators who have experience using the UNIX operating system. You should have at least two years of computer science experience or training. You should also be familiar with the processor hardware in your system. If you are a new user who will not be involved in computer operations, refer to the *UNIX System User Guide.*

Procedures and examples are given for starting up your system (booting and powering), changing run levels (that is, single user and multiuser), saving and restoring files, bringing down the system in an orderly manner, and restoring the system after a crash. The procedures and descriptions in the first four chapters of this guide are common to all processors running the UNIX operating system. Following these four chapters, you will find separate chapters containing specific information for each of the processors listed above. You should always consult the chapter for your processor before performing any of the procedures in this guide.

Most of the operations you do will involve the system console. All messages to the operator and input from the operator are via the system console. You will be using the system console in one of three modes:

- Console - The UNIX operating system is halted. Only console commands from the system console will be accepted. See the chapter concerned with operations on your processor for console descriptions and commands.

---

* PDP and VAX are trademarks of Digital Equipment Corporation.

† UNIX is a trademark of AT&T Bell Laboratories.

- Single user - The UNIX operating system is executing. The commands you enter on the system console are UNIX system commands. In single user mode you are always super-user. When the system is halted or in single user mode, the console is the only interface to the system, unless you specifically change the configuration so that another terminal acts as a console.

- Multiuser - The UNIX operating system is executing. The system console (and any other configured terminal) is treated as a normal user terminal.

In halt mode or single user mode, the console will not be treated as a login terminal (therefore, you are super-user). When you change the system to multiuser mode, a login message will appear on the console. You must provide a login and password at this point in order to use the console. Normally you should log in as **root**. Here, it must be mentioned that the login you (the operator) use is a local decision. In fact, your system administrator may configure your system so that it is not even necessary for you to log in after changing to multiuser. Chapters 3 and 4 describe normal daily maintenance requirements and provide examples of normal operations (not including local procedures). For more information on the console (for example, set-up procedures), consult your console terminal owner's manual.

Throughout this document, each reference of the form *name*(1M), *name*(7), or *name*(8) refers to entries in the *UNIX System Administrator Reference Manual.* All other references of the form *name*(N), where **N** is a number possibly followed by a letter, refer to entries in section **N** of the *UNIX System User Reference Manual* or *UNIX System Programmer Reference Manual.*

Throughout this document, the following notation is used for computer input/output:

1. Special characters are enclosed in <> (for example, when you see <**ctl**> read this as the "control" key and <**cr**> as the "carriage return" key).

2. Items within []s are optional.

3. You should type in literally any command field that appears boldface (a keyword).

4. You should substitute with the appropriate information any command field that appears in italics.

5. All commands (UNIX system or console commands) should be terminated with a carriage return.

Because this guide is intended to be as general as possible, no installation-specific information has been included. Also, some operations may vary according to local procedures. It is suggested that you add specific information about:

- Hardware configuration
- Software configuration of administrative files
- Data set configuration
- Specific logging and record-keeping practices
- Contacts for hardware and software problems
- Site-dependent diagnostic procedures.

Chapter 2

START-UP PROCEDURES

INTRODUCTION

This chapter describes how to start up your UNIX system. Starting the system may simply mean booting if the system has been shut-down routinely or if a crash has occurred. Or, it may be necessary to power up the processor and peripherals such as disk drives if they have been powered down. See the chapter *OPERATOR DUTIES* for shutting down procedures. The final step when starting up the system is the **boot**. The boot procedure loads a copy of the UNIX operating system from disk (or tape) into memory and executes it. You will need to reboot the UNIX operating system when one of the following conditions occur: system crash or restart; loading of a new software release; or updating of the software release.

The boot will typically put your system into the single user " run level" awaiting your commands. The UNIX operating system uses 'init' to automatically enter the final run level when properly configured by your system administrator. Run levels are discussed in the *SINGLE USER AND MULTIUSER* chapter of this guide. Normally, run level 's' indicates single user and '2' indicates multiuser. For more information on 'init' refer to *init*(1M) in the *UNIX System Administrator Reference Manual, inittab*(4) in the *UNIX System Programmer Reference Manual,* or consult your local UNIX system administrator.

See the chapter concerned with operations on your processor for powering and booting procedures.

Chapter 3

SINGLE USER AND MULTIUSER

INTRODUCTION

There are two main modes of operation of a UNIX operating system: single user (level S) and multiuser (level 2). The run level has 8 possible values: 0-6, and s (or S). Single user is always s or S. Although multiuser is normally level 2, your system administrator can configure the **/etc/inittab** file to run multiuser at any level from 0 to 6. Furthermore, the **inittab** file can be configured so that certain procedures are followed automatically only the first time a certain run level is entered. For example, normally you will be asked to verify date and file systems the first time you change your system to multiuser. This is caused by an entry in the **inittab** file. Subsequent changes in run level will not perform this procedure automatically unless you specifically change the **inittab** file. For more information on 'init' refer to *init*(1M) in the *UNIX System Administrator Reference Manual, inittab*(4) in the *UNIX System Programmer Reference Manual,* or consult your local UNIX system administrator.

When in single user mode, all dial-up ports and hard-wired terminals are disabled and only the console terminal may interact with the processor. This mode of operation allows you to make necessary changes to the system without any other processing taking place. However, you will normally run the UNIX operating system in multiuser mode. Consult the chapter for your processor before proceeding with any of these procedures.

SINGLE USER ENVIRONMENT

In single user mode, you may type any available UNIX system command (followed by a <**cr**>). When the system has completed execution of the command, it will prompt with the "#" again on the next line. You use the single user environment primarily to do filesaves, system maintenance, modification, or repair operations. The typical

sequence of commands to change the system to multiuser mode is:

- **fsck**
- **telinit 2**

The fsck Command

The command **fsck** will interactively repair any damaged file systems that result from a crash of the operating system. You should also use it to ensure that the file systems are not damaged before going into multiuser mode or taking filesaves. Usually, you will want to respond "**yes**" to all the prompts; however, in the event of a system crash, the damage may be extensive enough to warrant recovery from a backup pack. The procedure for this is discussed under *FILESAVES* in chapter 4. See **fsck** in the *UNIX System Administrator Reference Manual* for details on the various options available and the *UNIX System Administrator Guide.* for a description of all the different errors that can occur.

An example of a check of a consistent file system is illustrated below:

```
# fsck /dev/rdsk/6s1
/dev/rdsk/6s1
File System: usr Volume: p0603
** Phase 1 - Check Blocks and Sizes
** Phase 2 - Check Pathnames
** Phase 3 - Check Connectivity
** Phase 4 - Check Reference Counts
** Phase 5 - Check Free List
2441 files 16547 blocks 31889 free
#
```

A file system that has been damaged can be repaired as shown below. The **y** is your response. When checking a file system, you can avoid the questions asked by **fsck** concerning inconsistencies found by using the **y** option. This option will automatically attempt repairs as though you answered **yes** to the questions. Use this with caution—the corrections usually involve some data loss. If you decide to interactively repair the file system, then follow the example below:

```
# fsck /dev/rdsk/6s0
```

The UNIX operating system responds

```
# /dev/rdsk/6s0
File System: fs1 Volume: p0603
** Phase 1 - Check Blocks and Sizes
POSSIBLE FILE SIZE ERROR I=2500
** Phase 2 - Check Pathnames
** Phase 3 - Check Connectivity
** Phase 4 - Check Reference Counts
UNREF FILE I=2500  OWNER=255 MODE=100755
SIZE=0 MTIME=Dec 31 19830 1983
CLEAR? y
** Phase 5 - Check Free List
2441 files 16547 blocks 889 free
***** FILE SYSTEM WAS MODIFIED *****
#
```

All mountable file systems should be listed in the file **/etc/checklist** which **fsck** uses, and you should check these file systems each time the system is rebooted.

A faster alternative to using **fsck** is **checkall**. The **checkall** command uses **dfsck** (a front end for **fsck**) to simultaneously check two file systems in different disk drives. Therefore, you can check file systems faster with **checkall** than you can with **fsck**. Included in **checkall** are the file system names that normally appear in **/etc/checklist** (see **checkall** in the *UNIX System User Reference Manual*).

WARNING: Never execute fsck on a mounted file system; it will have a bad effect since you are repairing only the physical disk. The only exception to this is the root file system which is always mounted.

An example of repairing the **root** file system follows:

```
# fsck /dev/dsk/0s0
/dev/dsk/0s0
File System: root Volume: p0001
** Phase 1 - Check Blocks and Sizes
POSSIBLE FILE SIZE ERROR I=416
POSSIBLE FILE SIZE ERROR I=610
POSSIBLE FILE SIZE ERROR I=614
POSSIBLE FILE SIZE ERROR I=618
POSSIBLE FILE SIZE ERROR I=625
** Phase 2 - Check Pathnames
```

```
** Phase 3 - Check Connectivity
** Phase 4 - Check Reference Counts
UNREF FILE I=416  OWNER=uucp MODE=100400
SIZE=0 MTIME=Nov 20 16:23 1983
CLEAR? y
UNREF FILE I=610  OWNER=csw MODE=100400
SIZE=0 MTIME=Nov 20 16:26 1983
CLEAR? y
UNREF FILE I=625  OWNER=cath MODE=100400
SIZE=0 MTIME=Nov 20 16:26 1983
CLEAR? y
FREE INODE COUNT WRONG IN SUPERBLK
FIX? y
** Phase 5 - Check Free List
1 DUP BLKS IN FREE LIST
BAD FREE LIST
SALVAGE? y
** Phase 6 - Salvage Free List
585 files 5463 blocks 4223 free
***** BOOT UNIX (NO SYNC !) *****
#
```

At this time you must immediately halt the processor and then reboot the system (see the chapter concerned with operations on your processor for start-up procedures).

The telinit 2 Command

After you have checked the file systems, you may change the UNIX operating system to multiuser. Do this by entering the command **telinit 2**. This command activates processes that allow users to log in to the system, turn on the accounting and error logging, mount any indicated file systems, and start the **cron** and any indicated daemons. Depending upon the type of data set your site has, you may have to manually flip the toggles or pop the buttons on the data sets to allow users to log in.

MULTIUSER ENVIRONMENT

There are two ways to get to this level: by typing **telinit 2**; or, specifying a run level of 2 after the boot. Users are permitted to access all mounted file systems and execute all available commands. In this mode, you can perform file restore procedures and take periodic status checks of the system. Some of these periodic status checks can include:

- A check of free blocks (**df**) remaining on all mounted file systems to ensure that a file system does not run out of space.
- A check on rje (**rjestat**).
- A check on **mail** to root or whatever login receives requests for file restores.
- A check on the number of users on the system (**who**).
- A check of all running processes (**ps -eaf** or **whodo**) to determine if there is some process using an abnormally large amount of CPU time.

If your site has other run levels defined, you can use the **telinit** command to change to those run levels. Finally, to change a multiuser system to single user, refer to *SYSTEM SHUTDOWN* in chapter 4.

Chapter 4

OPERATOR DUTIES

INTRODUCTION

This chapter is meant to serve as a guide for your normal duties as a computer operator. These descriptions do not represent what your job duties are; they merely outline the general procedures necessary to ensure that the system operates properly. Consult the chapter for your processor before proceeding with any of these procedures.

FILESAVES

Unless timely copies of the file systems are saved, a major system crash could devastate your user community. The user files could be destroyed or become inaccessible.

You should take daily filesaves. Should the system crash and result in the loss of files, then, at most, only a day's work will be lost. If your last filesave (or backup) was a week ago, then even after restoring the file any changes made since that backup will be lost.

There are two main ways you can do filesaves: by disk and by tape. Most sites utilize **volcopy** to save files. See **volcopy** in the *UNIX System Administrator Reference Manual* for more information on the options available and the use of this command. You should normally do your file saving while in single user mode, with the file system unmounted, to preclude any file system activity and subsequent damage on the saved copy. Also, to ensure system buffers are flushed and file systems are up-to-date, the **sync** command should be executed before filesaves.

Normally the filesave procedure is automated by the system administrator. Your administrator may have created a " shell script" to perform the filesave as part of your site's local operation. Daily filesaves usually are made on disk; whereas, a weekly filesave, for example, would be more efficiently made on tape. Tape saves are

necessary for long term storage or for regular saves if you do not have a spare disk drive. Tapes may be previously labeled, or may be labeled by the **volcopy** command. Your administrator may have created separate shell scripts for disk and tape saves (that is, incorporating the procedures that follow).

You must have at least two (2) disk drives, one of them a spare, for the following procedures. For ease of mapping, file systems are normally saved in the same sections on the backup pack as they exist on the working pack. This is imperative if you ever need to boot from a backup version of **root**. It is required that the **root** file system reside on section **0** of the pack. Your responses are indicated in bold type.

Saving the Root File System on Disk

In this example, the **root** file system will be saved on drive **3**, volume *S3B002* (whatever volume name you use should match the external label sticker).

1. Mount the disk pack to contain the file-save on drive 3.

2. Spin up the drive.

3. Enter the commands:

```
# sync
# fsck /dev/dsk/0s0
# volcopy root /dev/rdsk/0s0 S3B001 /dev/rdsk/3s0 S3B002
```

   to copy the root file system from drive 0 section 0 to drive 3 section 0. The following messages should appear:

```
From: /dev/rdsk/0s0, to: /dev/rdsk/3s0? (DEL if wrong)
END: 23000 blocks.
#
```

   If the **from** and **to** file systems are correct, wait for the prompt; otherwise, press the **DEL** key on the terminal to abort the copy.

4. Do step 3 for all the sections of the disk that are to be copied.

5. Spin down drive 3 and remove the pack.

In the above procedure, **fsck** in step 3 will ask you to concur with any repairs necessary before attempting them. If you respond **no**, no action will be taken and **fsck** will continue. Also, **volcopy** will verify the label information on the **to** and **from** file system (for example, pack number, file system name, date last modified). You will be asked to override inconsistencies before the copy will proceed. For example:

```
# volcopy root /dev/rdsk/0s0 p0001 /dev/rdsk/3s0 p0105
arg.(p0105) doesn't agree with to vol.()
Type 'y' to override: y
warning! from fs(root) differs from to fs()
Type 'y' to override: y
From: /dev/rdsk/0s0, to: /dev/rdsk/3s0? (DEL if wrong)
END: 23000 blocks.
#
```

Note: In this example, the **to** section is unlabeled, as indicated by the null volume and file system fields. For more information see **volcopy** in the *UNIX System Administrator Reference Manual.*

Saving the User File System on Disk

In this next example, the **usr** file system, mounted on drive 6, will be saved on drive 3, volume *p0603.*

1. Mount the disk pack to contain the file-save on drive 3.

2. Spin up the drive.

3. Enter the commands:

```
# sync
# umount /dev/dsk/6s1
# fsck /dev/rdsk/6s1
# volcopy usr /dev/rdsk/6s1 p0001 /dev/rdsk/3s1 p0603
```

   to copy the usr file system from drive 6 section 1 to drive 3 section 1. The following messages should appear.

```
From: /dev/rdsk/6s1, to: /dev/rdsk/3s1? (DEL if wrong)
END: 23000 blocks.
#
```

If the **from** and **to** file systems are correct, wait for the prompt; otherwise, press the **DEL** key on the terminal to abort the copy.

4. Do step 3 for all the sections of the disk that are to be copied.

5. Spin down drive 3 and remove the pack.

Saving the User File System on Tape

In this example, the **usr** file system is saved on tape volume *t0001*, mounted on transport 0. The **labelit** command is used to label the tape before the copy. You should place an external paper label on the outside of the reel carrying the same information as is written in the tape header label. The external label should also indicate the sequence number of the tape, if it is from a set (multi-reel volume) for the file system. Note the use of the **-n** option to **labelit**. Unless this option is used on an unlabeled tape, the program will scan the entire reel looking for a label to change before it rewinds and labels the beginning. This can be very time-consuming on 2400-foot reels.

You can store approximately 65,000 blocks of a file system on a 2400-foot tape using **volcopy** and recording at 1600 bpi. You may specify the size and type of tape in the volcopy command, or you can let the system prompt for the information as shown. In the example that follows, the file system requires two reels. Although this example uses only one drive, you can have both reels mounted on different drives. In that case, when the first has finished, you would simply enter the name of the second drive when asked.

1. Load the tape in transport 0, and label it:

```
# labelit /dev/rmt/0m usr t0001 -n
Skipping label check!
NEW fsname = usr, NEW volume = t0001 -- DEL if wrong!!
#
```

2. Enter the following commands:

```
# sync
# umount /dev/dsk/0s1
# fsck -y /dev/rdsk/0s1
# volcopy usr /dev/rdsk/0s1 p0001 /dev/rmt/0m t0001
Enter size of reel in feet for <t0001>:  2400

Reel t0001, 2400 feet, 1600 BPI
You will need 2 reels.
(The same size and density is expected for all reels)
From: /dev/rdsk/0s1, to: /dev/rmt/0m? (DEL if wrong)

Writing REEL 1 of 2, VOL = t0001
Changing drives? (type RETURN for no, '/dev/rmt/_' for yes: <cr>

Mount tape 2
Type volume-ID when ready:  t0002
Cannot read header        (This tape has not been labeled!)
Type y to override: y
Volume is <garbage>, not <t0002>.
Want to override? y

Writing REEL 2 of 2, VOL = t0002
END: 90000 blocks.
#
```

FILE RESTORES

Restoring from Disk

When a request is made to restore a file from a backup pack, you should first locate that pack and determine on which section the requested file system resides. Then at the console terminal, log in to the system as **root** and proceed as the example illustrates. Following is the procedure for restoring the file **/usr/adm/acct/sum/tacct** from a previous backup pack. For this example, drive **4** is a spare drive and **/usr** is on section **1** of the backup pack.

1. Place backup pack on drive 4.

2. Spin up the drive.

3. Enter the command **mount /dev/dsk/4s1 /bck -r**. This will mount the backup file system as **/bck** read-only. The following message should appear.

```
WARNING!! - mounting <usr> as </bck>
```

4. Enter the command **ls -l /bck/adm/acct/sum/tacct**. This will verify the existence of the file and the identity of the owner. The following output will appear:

   ```
   -rw-rw-r-- 1 adm   bin  1932 Aug  9 14:27 /bck/adm/acct/sum/tacct
   ```

5. Enter the command:

   cp /bck/adm/acct/sum/tacct /usr/adm/acct/sum/tacct

   to copy the file from the backup to the specified place.

6. Enter the command:

   chown adm /usr/adm/acct/sum/tacct

   to change the owner of the file.

7. Enter the command **umount /dev/dsk/4s1**. This will unmount the backup file system.

8. Spin down the spare drive and remove the backup pack.

When you perform a file restore, it is usually a good practice to **mail** a message to the user asking for the restore when you are finished. Also, to avoid confusion, your message should refer to the file using a full pathname. The procedure for this is:

```
# mail user
I have restored the file /usr/adm/acct/sum/tacct
from Friday's backup.
operator's initials
#
```

Restoring from Tape

If the file does not exist on any of the backup packs or if your installation does not perform disk filesaves, then you will have to recover the file from a tape save. It is assumed that you do your tape saves in the same manner as disk saves, that is, with **volcopy**. The subject of filesaves is discussed earlier in this chapter under

FILESAVES. In order to restore a file from tape, you must place the whole file system on a spare section of the disk. The backup tape version can then be accessed in the same way as a disk save. For this example, it is assumed that there are two small file systems stored on a single tape and that the **usr** file system is the second file on the tape. Also, it is assumed that section **5** of disk drive **0** is a spare section on that disk. The tape drive is already in service.

1. Mount tape on tape drive 0.

2. Enter the command **echo < /dev/mt/0mn**. This will space past the first file on the tape, with no rewind.

3. Enter:

   volcopy usr /dev/rmt/0m t0004 dev/rdsk/0s5 S3B003

   This will copy the file system from tape to the spare disk section. The following messages should appear:

   ```
   From: /dev/rmt/0m, to: /dev/rdsk/0s5? (DEL if wrong)
   END: 90000 blocks.
   ```

4. Enter the command **mount /dev/dsk/0s5 /bck -r**. This will mount the backup section. The following message should appear on the screen:

   ```
   WARNING!! - mounting: <usr> as </bck>
   ```

5. Enter the command **ls -l /bck/adm/acct/sum/tacct**. This will verify the existence of the file and identify the owner. The following output will appear:

   ```
   -rw-rw-r-- 1 adm   bin  1932 Aug  9 14:27 /bck/adm/acct/sum/tacct
   ```

6. Enter:

   cp /bck/adm/acct/sum/tacct /usr/adm/acct/sum/tacct

   This will copy the file to the specified place.

7. Enter the command

```
chown adm /usr/adm/acct/sum/tacct
```

to change the owner of the file.

8. Enter the command **umount /dev/dsk/0s5**. This will unmount the spare section.

Sometimes a file system is so large it requires more than one tape to store the contents. In this situation, you will follow the same procedure to restore a file as in the previous example. The volcopy command will prompt you for additional reels when necessary. In this example, the second reel has the wrong label. The **y** response overrides the inconsistency and the reel is read anyway.

1. Mount tape on tape drive 0.

2. Enter:

```
volcopy -bpi1600 -feet2400 usr /dev/rmt/0m t0004 dev/rdsk/0s5 S3B003
```

This will copy the file system from tape to the spare disk section. The following messages should appear:

```
Reel 1, 2400 feet, 1600 BPI
From: /dev/rmt/0m, to: /dev/rdsk/0s5? (DEL if wrong)

Reading REEL 1 of 3, VOL = 1
Changing drives? (type RETURN for no, '/dev/rmt/_' for yes: <cr>
Mount tape 2
Type volume-ID when ready: 2
Volume is <1>, not <2>.
Want to override? y

Reading REEL 2 of 3, VOL = 1
Fri Jul 29 12:00:02 EDT 1983

Changing drives? (type RETURN for no, '/dev/rmt/_' for yes: <cr>
Mount tape 3
Type volume-ID when ready: 3

Reading REEL 3 of 3, VOL = 3
END: 90000 blocks.
```

3. Enter the command **mount /dev/dsk/0s5 /bck -r**. This will mount the backup section. The following message should appear on the screen.

```
WARNING!! - mounting: <usr> as </bck>
```

4. Enter the command **ls -l /bck/adm/acct/sum/tacct**. This will verify the existence of the file and identify the owner. The following output will appear:

```
-rw-rw-r-- 1 adm   bin  1932 Aug  9 14:27 /bck/adm/acct/sum/tacct
```

5. Enter:

```
cp /bck/adm/acct/sum/tacct /usr/adm/acct/sum/tacct
```

   This will copy the file to the specified place.

6. Enter the command

```
chown adm /usr/adm/acct/sum/tacct
```

   to change the owner of the file.

7. Enter the command **umount /dev/dsk/0s5**. This will unmount the spare section.

MESSAGE OF THE DAY

When a user logs into the system, part of the login procedure prints out a message of the day. This message can contain several lines of useful information to the user concerning scheduled down-time for hardware preventive maintenance (PM), clean-up messages for space-low file systems, or any other useful warnings to which users may need to be alerted. The trick to maintaining this file is to keep it short and to the point. A user does not want to wait ten minutes while eloquent and wordy dialogue is spewed from the terminal before he or she can begin working.

The contents of this message are stored in the file **/etc/motd**. You may change the contents of this file by using the UNIX system text editor. See *ed* in the *UNIX System User Reference Manual*. A sample of adding and deleting a line from this file is shown below.

```
# ed /etc/motd
```

```
26
p
9/23: Reboot at 5pm today.
d
a
9/24: Down for PM 1700-2100 on 9/30.
w
37
q
#
```

You can also remove the contents of the entire file (do not remove the file itself; it needs to exist so the login process can read it) by:

```
# cp /dev/null /etc/motd
#
```

SYSTEM SHUTDOWN

You will perform three distinct steps when bringing down your UNIX system. These steps must be performed in the indicated order, although it is not necessary to bring the system completely down for certain maintenance operations. For example, preventive maintenance (such as filesaves) must be done while in single user mode without halting the UNIX system. Whereas, repairing a hard fault would necessitate removing power completely. You should never remove power from a piece of equipment that is in service, and definitely do not power down the system until the UNIX operating system has been halted. To bring down the system:

- Run the shutdown program (changes a multiuser system to single user mode).
- Halt the UNIX program (the operating system).
- Remove power.

Shutdown Program

Whenever the system must be shut down, such as for filesaves or a reboot, you should run the program **/etc/shutdown**. The shutdown procedure is designed to gracefully turn off all processes and bring the system back to single user state with all buffers flushed.

You must be in the root directory (/) to use the shutdown program. You may specify the amount of grace period between sending a warning message out and actually shutting down. This grace period is the number of seconds of delay. For example, specifying a grace period of 300 will result in a 5-minute delay. You may also send your own message. A default message is sent to all logged-in users if you don't type your own. The following printout is an example of a typical shutdown sequence. Enter the following:

```
# cd /
# shutdown 300
```

Your shutdown procedure may vary slightly from the following, depending on how it is set up by your system administrator. The shutdown " script" may be modified by your administrator according to local procedures. A typical output is as follows:

```
SHUTDOWN PROGRAM

Thu Sep  1 18:51:58 EST 1983

Do you want to send your own message? (y or n): y
Type your message followed by <ctrl>d....

System coming down for filesaves!
Please log off.
<ctrl>d

System coming down for filesaves!
Please log off.
(waits for 5 minutes)
SYSTEM BEING BROUGHT DOWN NOW ! ! !

Busy out (push down) the appropriate
phone lines for this system.

Do you want to continue? (y or n):  y
Process accounting stopped.
```

```
Error logging stopped.
All currently running processes will now be killed.

Wait for 'INIT: SINGLE USER MODE' before halting.
```

If you executed the shutdown program while in single user mode, (which is neither useful nor recommended) the system will not respond with the 'INIT' message above.

At the completion of this program you can either halt the system (and reboot if necessary), power down, start the filesave routine or other preventive maintenance, or bring the system back to multiuser mode. To go to multiuser, type in **telinit 2**. See the chapter titled *SINGLE USER AND MULTIUSER* for more information on changing run level. See the chapter concerned with operations on your processor for power down procedures and booting procedures.

SYSTEM CRASH RECOVERY

An operating system is considered to have "crashed" when it halts itself without being asked to. The reason for the halt is often unknown and can be hardware failure or software related. It is important, for obvious reasons, to determine the nature of the crash so that it will not happen again. One way to do this is to take a dump of memory to disk (or tape) so that debugging programs can later decipher what processing was going on at the time the crash occurred.

See the chapter concerned with operations on your processor for recovery procedures.

Chapter 5

VAX-11/750 OPERATIONS

Information in this chapter has been condensed from Chapter 6 of the *VAX Hardware Handbook*, DEC, 1982-83. Also, refer to *11/750ops*(8) in the *UNIX System Administrator Reference Manual* for more information. The following notation will be used throughout this chapter for all computer input/output:

1. Special characters are enclosed in <> (for example, when you see <**ctl**> read this as the "control" key and <**cr**> as the "carriage return" key).

2. Items within []s are optional.

3. You should type in literally any command field that appears boldface (a keyword).

4. You should substitute with the appropriate information any command field that appears in italics.

5. All commands (UNIX system or console commands) should be terminated with a carriage return.

SYSTEM CONSOLE

The following are the standard console commands. The most abbreviated form is shown in parentheses.

<ctl>P — Puts the machine in Console I/O mode and halts the processor. A halt message is printed.

<ctl>U — Deletes the current input line.

<del> — Deletes the previous character.

(E)XAMINE *address*
Displays 8-digit hexadecimal address and its contents.

(D)EPOSIT *address data*
Enters data to address. Refer to the *VAX Hardware Handbook* for EXAMINE and DEPOSIT qualifiers.

(I)NITIALIZE Initializes CPU.

(H)ALT The HALT command is implemented on the VAX-11/750 for the sake of consistency with the VAX-11/780. It does not actually halt the CPU since the CPU must already be halted to respond to the command (see **<ctl>P**). It does reset the console defaults.

(S)TART *address*
Initializes CPU, enters address to PC, issues CONTINUE to CPU, and puts console into Program I/O mode.

(C)ONTINUE Allows the user to restart a halted program without altering the state of the machine.

START-UP

Powering Up

Following is the procedure for powering your VAX:

1. Power on and spin up each equipped disk drive. The **READY** light will light steadily when the disk is up to speed.

2. At the console, set power switch to **ON**.

3. At the processor front panel, set the **POWER ON ACTION** switch to the **HALT** position. See Figure 5-1.

4. Set key switch to **LOCAL**. This will restore power to the processor and cause the following to print on the console.

```
      % %
00000000    16
      >>>
```

5. The >>> prompt indicates the console is ready to accept commands. You may proceed with the boot.

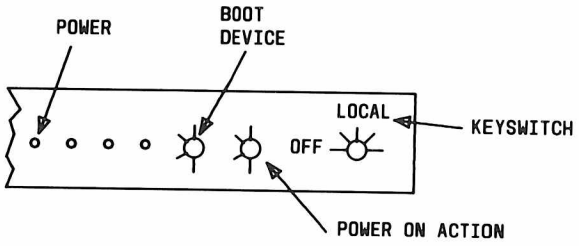

Figure 5-1. VAX-11/750 Front Panel

Booting from Disk

There is no floppy disk drive or floppy disk controller supplied with the console subsystem of the VAX-11/750, nor an LSI-11 microprocessor, as in the VAX-11/780. Instead, a TU58 tape cartridge is provided in the central processing unit (CPU) cabinet. You can use the TU58 to boot the system (that is, the bootstrap procedure can be stored in this cartridge). Refer to Chapter 6 of the *VAX Hardware Handbook*, Digital Equipment Corp., 1982-83 for details on powering and booting the VAX-11/750.

The first step in booting is to get the stand-alone shell running. When the system is first turned on, the console prompt >>> is printed. If you have shut down the UNIX operating system, but have not halted the processor (see *SYSTEM SHUTDOWN* in chapter 4), you must type **<ctl>p** to get into console mode. This also halts the CPU. With the system halted, any of the console commands may be executed as described under *SYSTEM CONSOLE*.

To boot the stand-alone shell from the default disk drive, type **B<cr>**. The following is an example of this operation as seen on the console, picking up after the **<ctl>p**:

```
>>>B<cr>
     %%

$$
```

There is a four-position switch on the front panel (**BOOT DEVICE**) that lets you select the device the boot command will boot from. Or, you can specify an argument in the boot command that selects the boot device superseding the switch-selected one. See Figure 5-2.

```
>>>B [ddcu] <cr>
```

CODE	DEVICE
DL	RL01/02
DB	RP04/05/06/07, RM03, & RM80
DD	DECTAPE II Cartridge (TU58)
DM	RK06/07

Figure 5-2. VAX-11/750 Boot Device Code Table

The **c** specifies the I/O channel adapter and **u** is one digit identifying the device drive number. For example,

```
>>>B DDA0 <cr>
```

will boot from the TU58.

The **$$** prompt indicates that the stand-alone shell is ready to accept commands. If you need to run stand-alone **fsck** or other stand-alone functions (see the *UNIX System User Reference Manual*), this is the time to do it. The commands have the form **/stand/program** where **program** can be any name from a limited list of UNIX operating system commands found in the directory **/stand**. To perform a file system consistency check, type:

```
$$ /stand/fsck /dev/dsk/0s0
```

When you are ready to bring up the UNIX operating system, type **unix<cr>**. The system will scan the **/etc/inittab** file to determine, among other things, which run level will be entered. If this file specifies a run level (or a default level is found) the system will enter the run level specified. Otherwise, do the following steps.

1. This message should appear on the console:

   ENTER RUN LEVEL (0-6, s, or S):

   Enter **2<cr>** to go to multiuser state or **s<cr>** to go to single user state.

2. If you requested multiuser in step 1, the system will ask you to verify the date. Then you will be asked if the file systems are to be checked. Finally, the following message will be printed on the console:

   Console Login:

   If you requested single user in step 1, the # prompt will be printed. In this case, typing **telinit 2** will change the operating system state to multiuser.

Booting from Tape (Installation)

You must type in the following program to read the first record on tape drive 0. Type <cr> at the end of each input line:

```
>>>H
>>>I
>>>D/G E 20000
>>>D/P 20000 F5508FD0
>>>D + D05000FF
>>>D + F308008F
>>>D + A0B45100
>>>D + D0421002
>>>D + 0000008F
>>>D + 8FD06180
>>>D + 80000100
>>>D + CFDE04A1
>>>D + CA53003A
>>>D + FFFE008F
```

```
>>>D + 8FC853FF
>>>D + 00000200
>>>D + 6053B053
>>>D + 8FB01B10
>>>D + 25CF0200
>>>D + 1DCFB400
>>>D + 018FB000
>>>D + 0014CFC0
>>>D + F46053B0
>>>D + BF002BCF
>>>D + 808FB300
>>>D + 1302A000
>>>D + 000005F8
>>>D + 029AC004
>>>D + 00080000
>>>D + 00000000
>>>D + 00000000
>>>D + 00000000
>>>D + 028C0000
>>>D + 000E0000
>>>D + 00010000
>>>D + 00000000
>>>S 20000                    Start tape load

00020055 06

>>>S 0                       Execute boot program
                             loaded from tape
```

From this point the loader initiates a question and answer sequence to control the remainder of the load process.

SHUTTING DOWN

Shutdown Program

The shutdown procedure is designed to gracefully turn off all processes and bring the system back to single user state with all buffers flushed. To do this you should execute **shutdown** as described in chapter 4. If **shutdown** is not successful, use the following sequence of commands.

```
killall
sync
init S
fsck            This is optional
```

In either case, you may halt the system by typing **<ctl>p**.

Powering Down

To power down, proceed as follows:

1. Run the shutdown sequence and halt the processor.
2. Power down all equipped disk drives (allow about 2 minutes for the disk to spin down before removing power).
3. On processor front panel, set key switch to **OFF**. See Figure 5-1.
4. At console (and all user terminals if practical), set power switch to **OFF**. The system is OFF.

CRASH RECOVERY

After a system crash, you should use the following procedure to get a system dump on tape:

1. Mount a tape with write ring and bring it on-line.
2. Halt the system and enter console mode with **<ctl>p**.
3. Issue the following command sequence, with each command followed by **<cr>**:

```
E/G 0       Keep typing E's until
            all registers have
            been examined: R0 through R15
S 400       Start execution at 400;
            i.e., dump to tape
```

4. Before returning to the UNIX operating system, execute the stand-alone **fsck**.

Chapter 6

VAX-11/780 OPERATIONS

Information in this chapter has been condensed from Chapter 14 of the *VAX Hardware Handbook*, DEC, 1982-83. Also, refer to *11/780ops*(8) in the *UNIX System Administrator Reference Manual* for more information. The following notation will be used throughout this chapter for all computer input/output:

1. Special characters are enclosed in <> (for example, when you see <**ctl**> read this as the "control" key and <**cr**> as the "carriage return" key).

2. Items within []s are optional.

3. You should type in literally any command field that appears boldface (a keyword).

4. You should substitute with the appropriate information any command field that appears in italics.

5. All commands (UNIX system or console commands) should be terminated with a carriage return.

SYSTEM CONSOLE

The following are the standard console commands. The most abbreviated form is shown in parentheses.

<ctl>P	Causes console to exit Program I/O mode (talking to the VAX-11/780 program). This does not halt the VAX CPU.
<ctl>U	Deletes the current input line.
<del>	Deletes the previous character.
<ctl>C	Interrupts printout.

(HE)LP — Prints "help" file of which this is a part.

(E)XAMINE *address*
Displays 8-digit hexadecimal address and its contents. See "help" file for qualifiers.

(D)EPOSIT *address data*
Enters data to address.

(I)NITIALIZE — Initializes CPU.

(U)NJAM — Unjams the SBI.

(SH)OW — Displays console and CPU state.

(H)ALT — Halts execution of VAX CPU instructions.

(S)TART *address*
Initializes CPU, enters address to PC, issues CONTINUE to CPU, and puts console into Program I/O mode.

(C)ONTINUE — Starts execution of VAX CPU instructions.

(SE)T (T)ERMINAL (P)ROGRAM
Puts console into Program I/O mode.

@file — Causes the named floppy file to be printed and executed.

WARNING: Only <ctl>p can be executed from Program I/O mode. It does not stop the VAX CPU from running. Only HALT can be executed while the VAX CPU is running and not in Program I/O mode; therefore, the sequence to stop the VAX-11/780 while running the UNIX operating system (Program I/O mode) is:
<ctl>p
>>>H<cr>

START-UP

Powering Up

Following is the procedure for powering your VAX:

1. Power on and spin up each equipped disk drive. The **READY** light will light steadily when the disk is up to speed.

2. At the console, set power switch to **ON**.

3. At the processor front panel, ensure both **AUTO RESTART** and **BOOT** switches are set to **OFF**. See Figure 6-1.

4. Set key switch to **LOCAL**. This will restore power to the processor.

5. The >>> prompt printed on the console indicates the system is ready to accept commands. You may proceed with the boot.

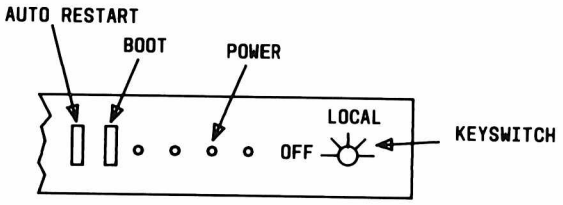

Figure 6-1. VAX-11/780 Front Panel

Booting from Disk

This procedure can be used only on a system with a floppy disk updated for use with the UNIX system. See the installation boot procedure for the update sequence.

The first step in booting is to get the stand-alone shell running: When the system is first turned on, the console prompt >>> is printed. If you have shut down the UNIX operating system, but have not halted the processor (see *SYSTEM SHUTDOWN* in chapter 4)

you must type **<ctl>p** to get into console mode and then **H<cr>** to halt the CPU. With the system halted, any of the console commands may be executed as described under *SYSTEM CONSOLE.*

To boot the stand-alone shell from the default disk drive, type **B<cr>**. Or, you can specify an argument in the boot command that selects the boot device superseding the default. For example, **B RP0** will boot from disk drive 0 and **B RP1** will boot from drive 1.

The following is an example of this operation, picking up after the **<ctl>p**:

```
>>>H<cr>
       HALTED AT nnnnnnnn

>>>B<cr>
       CPU HALTED
       INIT SEQ DONE
       HALT INST EXECUTED
       HALTED AT nnnnnnnn
       LOAD DONE, nnnnnnnn BYTES LOADED

$$
```

The **$$** prompt indicates that the stand-alone shell is ready to accept commands. If stand-alone **fsck** or other stand-alone functions are to be run (see the *UNIX System User Reference Manual*), this is the time to do it. The commands have the form **/stand/program** where **program** can be any name from a limited list of UNIX operating system commands found in the directory **/stand**. To perform a file system consistency check, type:

```
$$ /stand/fsck /dev/dsk/0s0
```

To bring up the UNIX system you must type **unix<cr>**. The system will scan the **/etc/inittab** file to determine, among other things, which run level will be entered. If this file specifies a run level (or a default level is found) the system will enter the run level specified. Otherwise, do the following steps.

1. This message should appear on the console:

   ENTER RUN LEVEL (0-6, s, or S):

   Enter **2**<cr> to go to multiuser state or **s**<cr> to go to single user state.

2. If you requested multiuser in step 1, the system will ask you to verify the date. Then you will be asked if the file systems are to be checked. Finally, the following message will be printed on the console:

   Console Login:

   If you requested single user in step 1, the # prompt will be printed. In this case, typing **telinit 2** will change the operating system state to multiuser.

Booting from Tape (Installation)

The floppy disk delivered with the VAX-11/780 does not have tape-boot capability. To initial load from tape, you must type in a program to read the first record on tape drive 0. If tape drive 0 is a TE16 or TU77 tape drive, use the first program provided below. For a TU78 tape drive, use the second program. Type <cr> at the end of each input line:

TE16 or TU77 Tape Drive

```
>>>H
>>>U
>>>I

      INIT SEQ DONE

>>>D 20000 20008FD0
>>>D + D0502001
>>>D + 3204A001
>>>D + C003C08F
>>>D + A0D40424
>>>D + 8FD00C
>>>D + C0800000
>>>D + 10A0FE00
>>>D + C007D0
>>>D + C039D004
>>>D + 400
```

```
>>>S 20000                    Start tape load
       HALT INST EXECUTED
       HALTED AT 0002002F

>>>S 0                        Execute boot program
                             loaded from tape
```

TU78 Tape Drive

```
>>>H
>>>U
>>>I

       INIT SEQ DONE

>>>D 20000 20008FD0
>>>D + D4502001
>>>D + 8FD00CA0
>>>D + 80000000
>>>D + 320800C0P
>>>D + A0FE008F
>>>D + C004D010
>>>D + 39320404
>>>D + 000400CP0
>>>S 20000                    Start tape load
       HALT INST EXECUTED
       HALTED AT 00020029

>>>S 0                        Execute boot program
                             loaded from tape
```

From this point the loader initiates a question and answer sequence to control the remainder of the load process.

Booting from Disk (Installation)

The floppy disk delivered with the VAX-11/780 does not have UNIX system disk-boot capability. To initial load from disk, you must type in the following program to read the first block on disk drive 0. Type <cr> at the end of each line:

```
>>>H
>>>LINK                     Save the following sequence on
                             disk (note the <<< prompt)
<<<H                         Halt processor
```

```
<<<U                    Unjam the SBI
<<<I                    Initialize the processor
<<<D/I 1 20003800       Register initialization
<<<D R0 0               Device type code
<<<D R1 8               NEXUS number of MBA
<<<D R2 0               drive number
<<<D R3 0               drive number
<<<D R4 0
<<<D R5 8               Software boot flags
<<<D FP 0               Set " no machine check expected"
<<<S 20003000           Start rom program
<<<WAIT DONE
<<<E SP                 Show address of
                        working memory +0X200
<<<L VMB.DS LE/S:@      Load primary bootstrap
<<<S @                  and start it
<<<<ctl>C               Exit LINK mode
>>>
```

You are now ready to boot the UNIX operating system. Each time it is necessary to boot (or reboot) the UNIX operating system, simply type **>>>P<cr>** after the >>> prompt. The console should echo each command in the floppy link file (program just saved on disk) as it is executed.

Enter **P<cr>** and wait for the **$$** prompt. The **$$** indicates that the stand-alone shell is ready to accept commands. If stand-alone **fsck** or other stand-alone functions are to be run (see the *UNIX System User Reference Manual*), this is the time to do it. The commands have the form **/stand/program** where **program** can be any name from a limited list of UNIX operating system commands found in the directory **/stand**. To perform a file system consistency check, type:

```
$$ /stand/fsck /dev/dsk/0s0
```

To bring up the UNIX system you must type **unix<cr>**. The system will scan the **/etc/inittab** file to determine, among other things, which run level will be entered. If this file specifies a run level (or a default level is found) the system will enter the run level specified. Otherwise, do the following steps:

1. This message should appear on the console:

   ```
   ENTER RUN LEVEL (0-6, s, or S):
   ```

   Enter **2**<cr> to go to multiuser state or **s**<cr> to go to single user state.

   If you need to update the floppy disk for the UNIX system, go to single user state and enter the following:

   ```
   #cd /stand/conflp
    sh update
   ```

   Update prints commentary during the update operation indicating the files that are being replaced or added. Finally, a new table of contents is printed and the available space is indicated.

2. If you requested multiuser in step 1, the system will ask you to verify the date. Then you will be asked if the file systems are to be checked. Finally, the following message will be printed on the console:

   ```
   Console Login:
   ```

   If you requested single user in step 1, the # prompt will be printed. In this case, typing **telinit 2** will change the operating system state to multiuser.

SHUTTING DOWN

Shutdown program

The shutdown procedure is designed to gracefully turn off all processes and bring the system back to single user state with all buffers flushed. To do this you should execute **shutdown** as described in chapter 4. If **shutdown** is not successful, use the following sequence of commands.

```
killall
sync
init S
fsck            This is optional
```

You may then halt the system by typing the **<ctl>p** and **H<cr>** sequence.

On occasion, the UNIBUS or its devices fail in such a manner as to flood the console with error messages and suspend operations on UNIBUS devices. Under these conditions, you may be able to bring the system down gracefully, from an internal point of view, by inhibiting UNIBUS interrupts and running a normal shutdown. The following sequence can be executed:

```
<ctl>p
>>>H
>>>E 20006004      Look at UBA control register
>>>D * 1           Clear the UBA
>>>C               Return to the UNIX operating system
```

You should now be able to run a normal shutdown sequence and halt the processor.

Power Down

To power down, proceed as follows:

1. Run the shutdown sequence and halt the processor.
2. Power down all equipped disk drives (allow about 2 minutes for the disk to spin down before removing power).
3. On processor front panel, set key switch to **OFF**. See Figure 6-1.
4. At console (and all user terminals if practical), set power switch to **OFF**. The system is OFF.

CRASH RECOVERY

After a system crash, the following procedure should be used to get a system dump on tape:

1. Mount a tape with write ring and bring it on-line.
2. Enter console mode with <**ctl**>**p**.
3. After the >>> prompt, halt the system with **H**<**cr**>.
4. Issue the following command sequence, with each command followed by <**cr**>:

E R0/N:F	*Examine R0 through R15*
E SP	*Get the stack pointer for the next command*
E/V @/N:3F	*Examine virtual memory beginning at the address from the previous instruction, and continuing for the next 63 locations; i.e., examine the stack*
S 400	*Start execution at 400; i.e., dump to tape*

5. Before returning to the UNIX operating system, execute the stand-alone **fsck**.

To read the dump tape into a file for examination by *crash* or *adb*, use *dd*. For example, the following will read the dump and create a file named core in the current directory:

```
dd if=/dev/rmt0 of=core bs=16b
```

Chapter 7

PDP-11/70 OPERATIONS

Booting Using the M9301-YC Loader

To bootstrap programs from a wide range of storage media, the PDP-11/70 has a dedicated diagnostic bootstrap loader called the M9301-YC. The M9301-YC contains hardware verification diagnostic routines and bootstrap loader routines in read-only memory (ROM).

The diagnostic portion tests the basic CPU to verify correct operation, then tests virtual memory from 001 000 to 157 776 and cache memory. Next, physical memory is tested as specified by console switches <15:12> (see Figure 7-1).

The bootstrap portion of the M9301-YC attempts to boot from the device and drive number specified by console switches <7:3> and <2:0>, respectively (see Figure 7-2). Having selected a boot device, the program will read a block of data into memory starting at virtual address 0; and then, will jump to virtual address 0. The physical address selection for the boot portion is the same as for the diagnostic portion. Excluding the RX11/RX01 floppy disk, bootstrap programs must fit in one block of 256 words, even though this program may read in more (depending on the device selected).

To start operation of the bootstrap loader:

1. Halt the CPU by depressing the **HALT** switch.
2. Set the **Address Display** select switch to **Console Physical**.
3. Set the Console Switch Register to **165 000**.
4. Depress the **Load Address** switch.
5. Reset the console switches to **0**.

6. Set switches <**15:12**> for the desired physical memory (normally 0).

7. Set switches <**7:3**> for the desired device (0 for default device).

8. Put the **HALT** switch in the **ENABLE** position.

9. Depress the **START** switch.

The diagnostic portion will then run followed by the boot from the selected device. This takes approximately three seconds. Any error during the diagnostic portion will cause the CPU to halt. If the CPU halts due to a cache error, pressing the **CONTINUE** switch causes the program to disable the cache and then proceed to the bootstrap section. If there is an error in reading the boot block, the program will do a RESET instruction, jump back to the memory test, and then attempt to boot again.

CONSOLE SWITCHES <15:12>	PHYSICAL ADDRESSES
00	00 000 000 - 00 157 776
01	00 200 000 - 00 357 776
02	00 400 000 - 00 557 776
03	00 600 000 - 00 757 776
04	01 000 000 - 01 157 776
05	01 200 000 - 01 357 776
06	01 400 000 - 01 557 776
07	01 600 000 - 01 757 776
10	02 000 000 - 02 157 776
11	02 200 000 - 02 357 776
12	02 400 000 - 02 557 776
13	02 600 000 - 02 757 776
14	03 000 000 - 03 157 776
15	03 200 000 - 03 357 776
16	03 400 000 - 03 557 776
17	03 600 000 - 03 757 776

Figure 7-1. PDP-11/70 Physical Memory Selection Table

CONSOLE SWITCHES <7:3>	DEVICE
00	Illegal
01	TM11/TU10 Magnetic tape
02	TC11/TU56 DECtape
03	RK11/RK05 Disk pack
04	RP11/RP03 Disk pack
05	Reserved
06	RH70/TU16 Magnetic tape
07	RH70/RP04 Disk pack
10	RH70/RS04 Fixed head disk
11	RX11/RX01 Diskette
12-37	Illegal

Figure 7-2. PDP-11/70 Device Selection Table